ID# 6025

TRAFFIC CAPACITY TESTING A WEB ENVIRONMENT WITH TRANSACTION BASED TOOLS

James F Brady
Capacity Planner for the State of Nevada
jfbrady@doit.nv.gov

A common server configuration that has emerged for supporting Web based applications is the Web Server, Application Server, and Database Server arrangement. This paper discusses a specific traffic capacity testing experience with this server combination for a Windows® environment using a transaction oriented load generator instead of a traditional virtual user script based tool. Both challenges encountered and insights gained are described.

1. Introduction

Often requirements dictate that a transaction based application be capacity tested before it is released into production using some type of traffic generation mechanism that simulates customer requests to a target system. A common server configuration that has emerged for supporting Web based applications is the Web Server, Application Server, and Database Server arrangement. This paper discusses a traffic capacity testing experience with this server configuration for a Windows environment using a transaction oriented load generator instead of a traditional virtual user script based tool. Both challenges encountered and insights gained are described.

The software being traffic capacity tested is a new application where the customer is doing the test because there is no performance data available from the vendor. The application is Web based with users logging on, performing queries of and updates to a large account database.

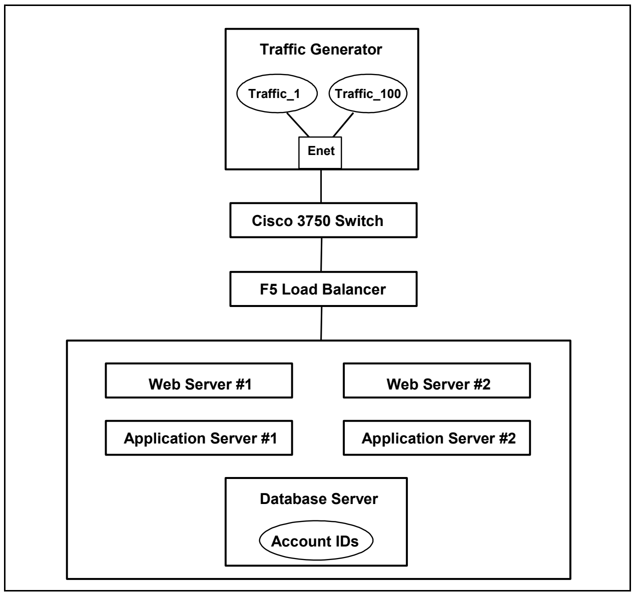
This is the initial capacity test of both the application software and the hardware configuration. Project scheduling constraints limited the test's scope to home page accesses, login events, and query activity. Follow-on tests are planned which expand functional coverage to include database updates and an increased set of queries.

The paper begins with a description of the traffic generation topology, some system configuration items, and the transaction traffic mix used. This information is followed by overall results, details in both graphical and tabular form, and a list of lessons learned along with some summary comments.

2. Traffic Generation Environment

Figure 1 shows the layout of the traffic generation environment including the Traffic Generator, a Cisco® Packet Switch, an F5® Load Balancer and five target servers. There are two Web Servers, two Application Servers, and a single Database Server shown. Although only one Database Server is depicted, there is a standby available which has been omitted since it did not participate in the testing process. The Traffic Generator is connected directly to the Cisco Packet Switch, eliminating any network latency.

Figure 1: Traffic Generation Topology



The traffic flow through this environment is as follows. The Traffic Generator's Traffic_1 through Traffic_100 processes create independent web requests that traverse the 100-megabit Ethernet connection on their way to the Cisco 3750 Switch which hands them off to the F5 Load Balancer. The Web Server selected by the load balancer packages and sends the request to the Application Server who produces an inquiry for the Database Server. The Database Server formulates an SQL request, retrieves the required data, and returns it to the Application Server. This data is forwarded up through the Web Server, F5 Load Balancer, and Cisco 3750 Switch until it reaches the Traffic Generator process making the request. This process analyzes the response to determine if it is correct and records the transaction response time.

Table 1 lists the configuration characteristics of each computing system in Figure 1, including the Traffic Generator. This table contains each system's name, description, CPU type, speed, and execution element count, along with the quantity of RAM and type of operating system used. The CPU "Execution Element" column depicts the number of processors displayed by the Windows Task Manager. The DB Server, for example, possesses four hyper-threaded processors resulting in eight "Execution Elements" being displayed by its Windows Task Manager.

Table 1: System Configuration

			CPU				
Item	Name	Description	Type	Speed (GHz)	Execution Elements	Ram (GB)	Operating System
1	TG	Traffic Generator	Pentium 4	3.4	2	2	Linux
2	WB1	Web Server	Xeon	3.2	4	4	Windows 3000
3	WB2	Web Server	Xeon	3.2	4	4	Windows 3000
4	AP1	Application Server	Xeon	3.4	4	8	Windows 3000
5	AP2	Application Server	Xeon	3.4	4	8	Windows 3000
6	DB	Database Server	Xeon	3.0	8	3.25	Windows 3000

The traffic generator, developed by this author, is transaction based with the one-hundred Figure 1 traffic processes running the Table 2 traffic mix. As shown, ten of the traffic processes request the application home page while the remaining ninety perform queries with half of the processes exercising Query 5. This query requests account data by randomizing over 10,000 account IDs. The ninety query users login one time at the beginning of each traffic run before repeatedly requesting their assigned URL. This transaction approach to traffic generation is applicable because the functionality is implemented with the connectionless HTTP internet protocol and sessions are managed at the application level using browser cookies [JECK04].

Table 2: Traffic Mix – Transaction

Item	Traffic Processes	Web Page Description	Web Action	Login First	Account IDs		URL
1	10	Home	GET	No	0		https://home.page.htm
2	10	Query 1	GET	Yes	0		https://query1.htm
3	10	Query 2	GET	Yes	0		https://query2.htm
4	10	Query 3	GET	Yes	0		https://query3.htm
5	10	Query 4	POST	Yes	1		https://query4.htm
6	50	Query 5	POST	Yes	10,000		https://query5.htm
Total	100						

Traffic generators that support connection oriented protocols usually implement a virtual user front end and state machine software to maintain session context between transaction sequences. This virtual user structure is often implemented for simpler connectionless situations as well. Table 3 is an example and lists five Vuser Types, each invoking a set internet transaction sequence with fixed or uniformly distributed delay times between transactions.

Table 3: Traffic Mix - Virtual User

	Delay	Vuser Type 1	Vuser Type 2	Vuser Type 3	Vuser Type 4	Vuser Type 5	Total
Transaction	Seconds	10%	15%	40%	10%	25%	100%
LOGIN	10	1	1	1	1	1	100%
Query1	15	1		1		1	75%
Query2	10		1		1	1	50%
Query3	20	1					10%
Query4	15		1				15%
Query5	20			1	1		50%
LOGOUT	10	1	1	1	1	1	100%
	Script Time	55	45	55	50	45	50.5

For a connectionless application, like the one analyzed in this paper, the virtual user structure may be overkill and the simpler transaction mechanism a better fit. Some of the beneficial characteristics of the transaction tool used here are:

1. It reports transaction request timing statistics to ensure offered traffic conforms to a "real world" random arrivals pattern.
2. It is structured to produce a consistent traffic mix during a capacity study.
3. It includes a set of analysis tools that support X-Y Plot construction of target server resource consumption levels as a function of transaction rate.

An X-Y Plot of traffic rate versus resource utilization can identify bottlenecks and imbalances for some resources by graphically determining if they have linear throughput characteristics. Resource utilizations with this functional property include CPU Usage, Disk I/O rates, and Packet rates. X-Y Plots of these counters are in the detailed analysis, Section 4, and Appendix A.

Determining the number of active users supported requires additional calculations, but the computations are a straightforward extrapolation from the transaction mix already established. For a detailed discussion of transaction based versus virtual user oriented traffic generation techniques see [BRAD06].

3. Traffic Capacity - Summary

The test procedure used for this application is to perform a series of 15 minute runs at increasing levels of traffic until resource saturation occurs. Traffic is generated in a random arrivals fashion where all processes draw their negative-exponentially distributed delay times from the same mean value during a run. Arrival patterns, response times, and resource consumption levels are recorded at each traffic increment. Server resource consumption statistics are gathered using a Windows PerfMon logging template set to sample every 30 seconds. Arrival events and response time values are recorded by the traffic generating software. There were nine 15 minute runs performed in this study before a specific resource bottleneck was found.

Table 4 contains a summary of the traffic capacity study results. It shows the maximum traffic rate achieved by the traffic generator is 4.42 web accesses per second with a mean response time of 383 milliseconds. There is a small amount of additional traffic being produced by approximately one dozen “hands on” users that is not reflected in the 4.42 Trans/sec but contributes to the resource consumption levels listed.

None of the resource utilization levels shown in Table 4 for CPU, Disk, Enet, or Memory are significant except for the Database Server’s memory at 657 Pages/sec. The paging rate actually reached an average of 2,120 Pages/sec for the 3.92 Trans/sec test run. Since the Database Server only has 3.25 GB of RAM available, additional memory appears to be required to eliminate this memory bottleneck. It is assumed the high paging rate is why the traffic generator consistently failed login attempts when trying to run above 4.42 Trans/sec. It is this author’s experience that any significant sustained paging rate on a system leads to performance and stability problems like those mentioned here.

These findings yield a recommendation to increase Database Server memory to 16 GB of RAM and rerun the traffic capacity test.

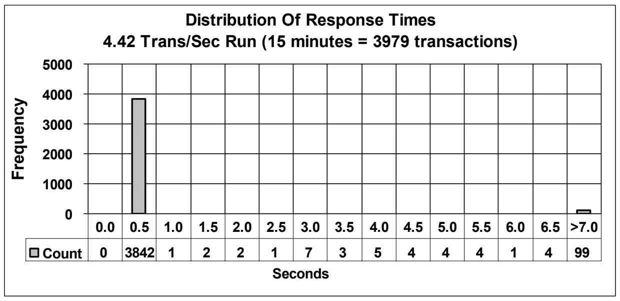
Table 4: Maximum Traffic Summary Stats

Traffic Rate and Response Time				
Computer	Traffic	Response Time (milliseconds)		
System	Trans/Sec	Median	Mean	95%
Traffic Gen	4.42	26	383	109

Computer Resource Consumption Levels				
	Computer Resources			
Computer	CPU	Disk	Enet	Memory
System	% Busy	I/Os /Sec	Packets/Sec	Pages/Sec
Traffic Gen	1%	0	216	0
Web Server #1	0%	1	162	0
Web Server #2	0%	1	155	0
App Server #1	1%	3	25	0
App Server #2	1%	5	208	0
DB Server	10%	290	1039	657

The response time statistics provided in Table 4 show the median response time to be an order of magnitude less than the arithmetic mean (26 ms Versus 383 ms), indicating a large number of very short response times are being averaged with a few very long ones. This is perhaps best illustrated by the response time distribution bar chart, Figure 2, that shows 3,842 of the 3,979 response times recorded are less than 500 ms but 99 of the remaining 137 exceed 7 seconds. The general breakdown of response times is that logins average around 6 seconds, web pages or cached account data retrievals take 30 ms, non-cached account data retrievals consume 60 ms, and the maximum response time is a 24.035 second login. These statistics and their associated graphical representation illustrate the misleading nature of service level averages within this test environment.

Figure 2: Distribution Of Response Times



	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0	5.5	6.0	6.5	>7.0
□ Count	0	3842	1	2	2	1	7	3	5	4	4	4	1	4	99

4. Traffic Capacity – Graphical View

The following graphs and supporting tables expand on the information listed in Table 4 by illustrating the reaction of the target servers and the traffic generator to incremental increases in traffic. Figure 3 shows traffic generator behavior as a function of transactions per second with Figure 4, Figure 5, and Figure 6 providing this information from a Web Server, Application Server, and Database Server perspective. Each of these figures includes response time graphs from a traffic generator orientation along with CPU Utilization, Disk I/O rate, Packet rate, and Memory Page rate plots. The table which follows each graph contains the data used to produce it along with details such as packet rates by direction and CPU utilization separated into total and system percentages.

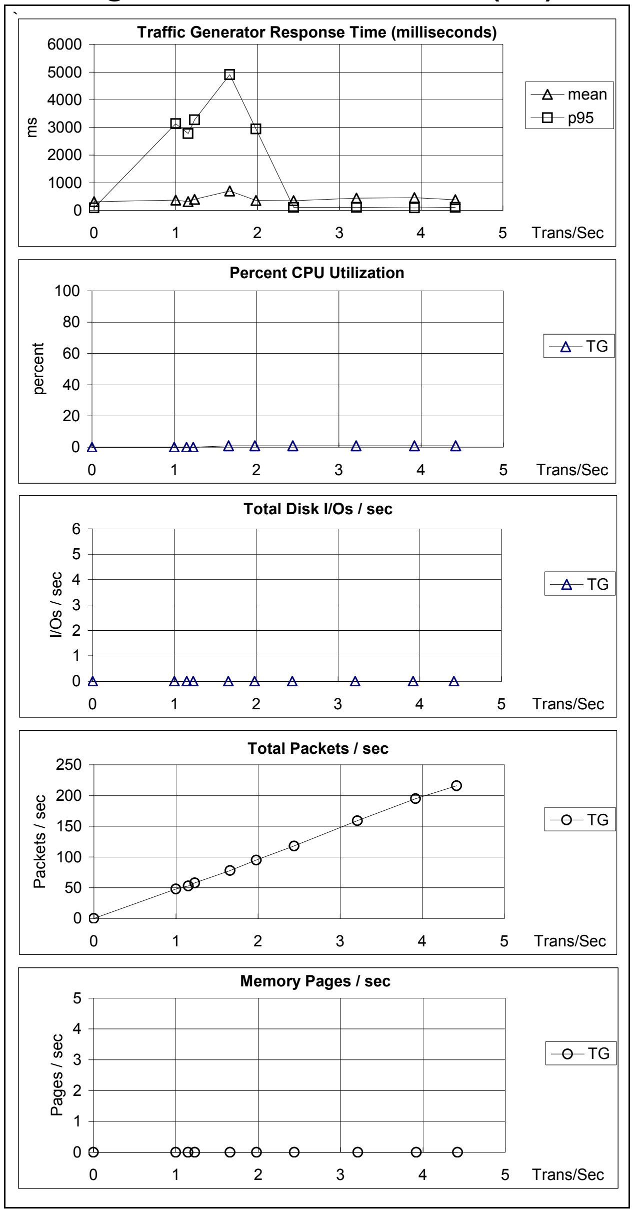
The following are comments and observations regarding these graphs and tables:

- All runs reflect 100% successful transactions with the browser timeout set to 30 seconds.
- All of the graphs are X-Y Plots where resource consumption is charted as a function of transaction rate.
- The Figure 3 packet rate is a linear function of transaction rate and would statistically match the sum of the packet rates plotted in Figure 4 if the traffic generator was the only traffic source. Unfortunately, the transaction rate and mix produced by the “hands on” users is unknown and although small, alters this functional relationship.
- Since all traffic generator processes are set up to operate independently at an overall constant traffic rate, they should produce Negative-Exponentially distributed inter-arrival

times. Compliance with this traffic pattern can be determined by comparing the inter-arrival time mean to its standard deviation because these two statistics are equal for Negative-Exponentially distributed data. The Arrival Summary Statistics table at the bottom of Figure 3 lists the mean and standard deviation of the inter-arrival times for 4.42 Trans/sec as 225 ms and 229 ms indicating near equality and conformance to the Negative-Exponential. For a discussion of this statistical property and its traffic generation significance see [BRAD04] and [BRAD06].

- The high p95 (95th percentile) response time values at low transaction rates are caused by the large login response times. This happens because each process performs one login per run and logins are a larger percentage of the traffic for low volume runs than for high volume runs. An analysis is planned to identify why login response times are so long, even when the traffic rate is low.
- CPU utilization, Disk I/O rates, and Packet rates are not currently bottlenecks for any of the servers analyzed. Memory Pages/Sec for the Database Server shown in Figure 6 is the obvious transaction rate constraint.
- The packet rates contained in Figure 4 for WB1 and WB2 are approximately the same, indicating that the F5 Load Balancer is working correctly.
- The inconsistent Disk I/O and Packet rate Versus Transaction rate pattern for the Application Servers, Figure 5, seems to indicate the work isn't well balanced across AP1 and AP2. This imbalance will be closely monitored as testing progresses.
- The erratic Disk I/O rate and Packet rate Versus Transaction rate pattern for the Database Server, Figure 6, likely results from that server's memory constraint, but to eliminate speculation, memory should be added and a follow-up test conducted.
- Traffic generator resource consumption data and target system response time information were collected and analyzed using a combination of standard operating system tools and tools developed by this author.
- A pie chart showing the proportion of CPU time consumed by the key processes is usually produced for each server but CPU utilization was too low when the bottleneck was reached to permit construction of this chart. Appendix A contains a process proportions pie chart example to show how one is developed and interpreted.

Figure 3: Traffic Generator (TG)



TG: Response Time, CPU, Disk I/O, Packets, Memory

run	tps	rt mean	rt p95	cpu sys	cpu tot	dsk r/s	dsk rw/s	pkt r/s	pkt rs/s	mem pg/s
1	0.00	315	86	0	0	0	0	0	0	0
2	1.00	372	3133	0	0	0	0	29	48	0
3	1.15	315	2778	0	0	0	0	32	53	0
4	1.23	398	3269	0	0	0	0	35	58	0
5	1.66	701	4908	0	1	0	0	47	78	0
6	1.98	356	2940	0	1	0	0	57	95	0
7	2.44	351	112	0	1	0	0	71	118	0
8	3.21	437	107	0	1	0	0	96	159	0
9	3.92	462	86	0	1	0	0	117	195	0
10	4.42	383	109	0	1	0	0	130	216	0

Arrival Summary Statistics

run	pass	n	tps	median	mean	sdev	p90	p95	min	max
1	1000	900	1.00	731	998	974	2153	2813	0	8134
2	900	1034	1.15	596	866	896	2101	2668	0	7587
3	800	1104	1.23	573	813	799	1886	2386	0	5586
4	600	1489	1.66	430	602	601	1416	1855	0	6029
5	500	1779	1.98	366	505	507	1173	1470	0	4727
6	400	2193	2.44	286	409	417	935	1216	0	3627
7	300	2889	3.21	220	310	299	703	940	0	2107
8	250	3531	3.92	174	254	252	584	770	0	1767
9	220	3978	4.42	155	225	229	524	692	0	1899

Response Time Summary Statistics

run	pass	n	tps	median	mean	sdev	p90	p95	min	max
1	1000	901	1.00	32	372	1053	210	3133	9	6197
2	900	1035	1.15	26	315	960	99	2778	9	6982
3	800	1105	1.23	50	398	1441	86	3269	9	26264
4	600	1490	1.66	24	701	2904	76	4908	8	27842
5	500	1780	1.98	27	356	1547	69	2940	9	18602
6	400	2194	2.44	23	351	1716	59	112	8	19408
7	300	2890	3.21	25	437	2213	53	107	8	24261
8	250	3532	3.92	23	462	2502	51	86	8	28257
9	220	3979	4.42	26	383	2117	57	109	8	24035

Figure 4: Web Servers (WB1 & WB2)

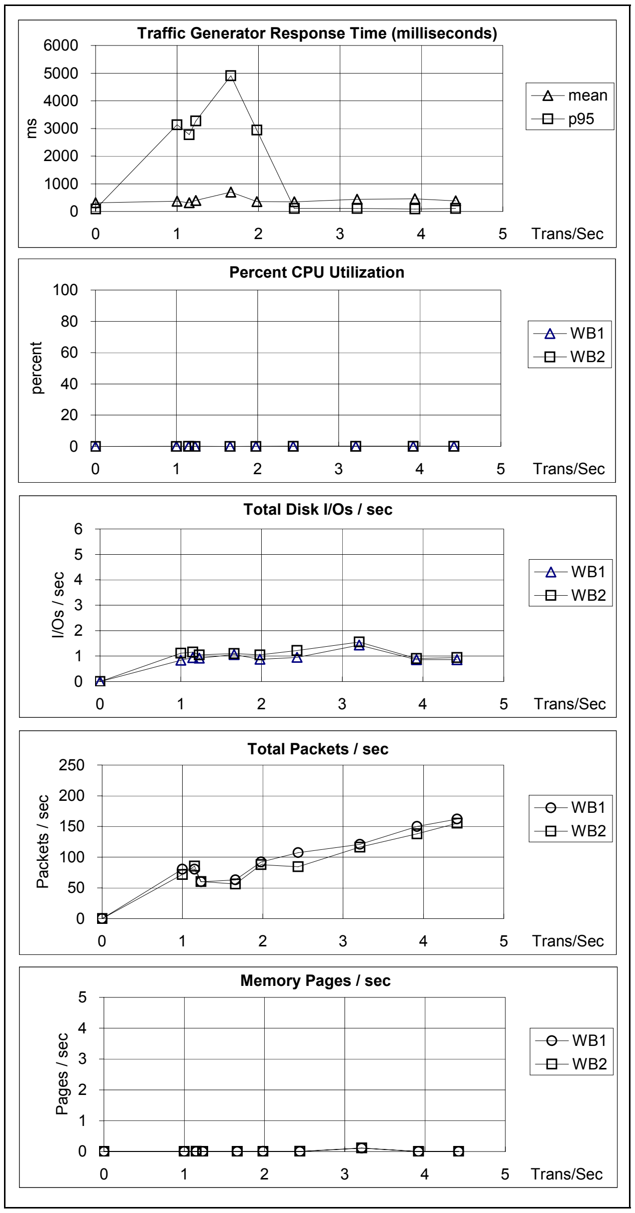
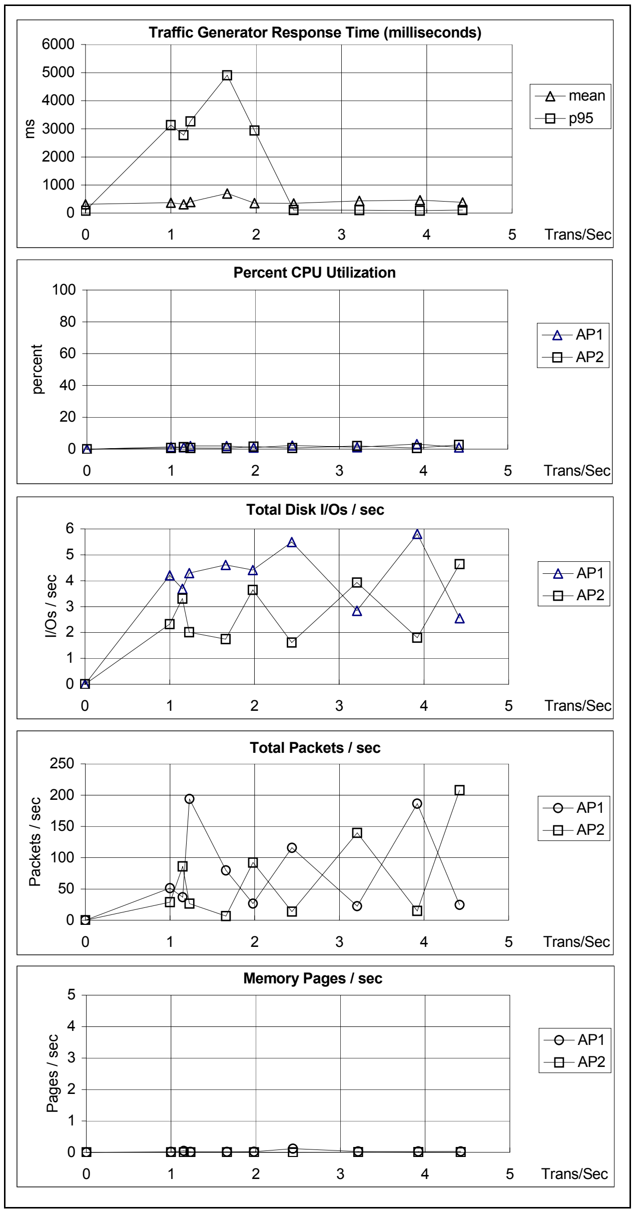


Figure 5: APP Servers (AP1 & AP2)



WB1: CPU, Disk I/O, Packets, Memory

run	tps	rt mean	rt p95	cpu sys	cpu tot	dsk r/s	dsk rw/s	pkt r/s	pkt rs/s	mem pg/s
1	0.00	315	86	0	0	0	0	0	0	0
2	1.00	372	3133	0	0	0	1	50	81	0
3	1.15	315	2778	0	0	0	1	48	81	0
4	1.23	398	3269	0	0	0	1	36	60	0
5	1.66	701	4908	0	0	0	1	38	63	0
6	1.98	356	2940	0	0	0	1	54	92	0
7	2.44	351	112	0	0	0	1	63	107	0
8	3.21	437	107	0	0	0	1	73	121	0
9	3.92	462	86	0	0	0	1	90	150	0
10	4.42	383	109	0	0	0	1	97	162	0

WB2: CPU, Disk I/O, Packets, Memory

run	tps	rt mean	rt p95	cpu sys	cpu tot	dsk r/s	dsk_rw/s	pkt r/s	pkt rs/s	mem pg/s
1	0.00	315	86	0	0	0	0	0	0	0
2	1.00	372	3133	0	0	0	1	44	72	0
3	1.15	315	2778	0	0	0	1	52	86	0
4	1.23	398	3269	0	0	0	1	36	60	0
5	1.66	701	4908	0	0	0	1	33	56	0
6	1.98	356	2940	0	0	0	1	51	88	0
7	2.44	351	112	0	0	0	1	51	84	0
8	3.21	437	107	0	0	0	2	70	116	0
9	3.92	462	86	0	0	0	1	83	138	0
10	4.42	383	109	0	0	0	1	93	155	0

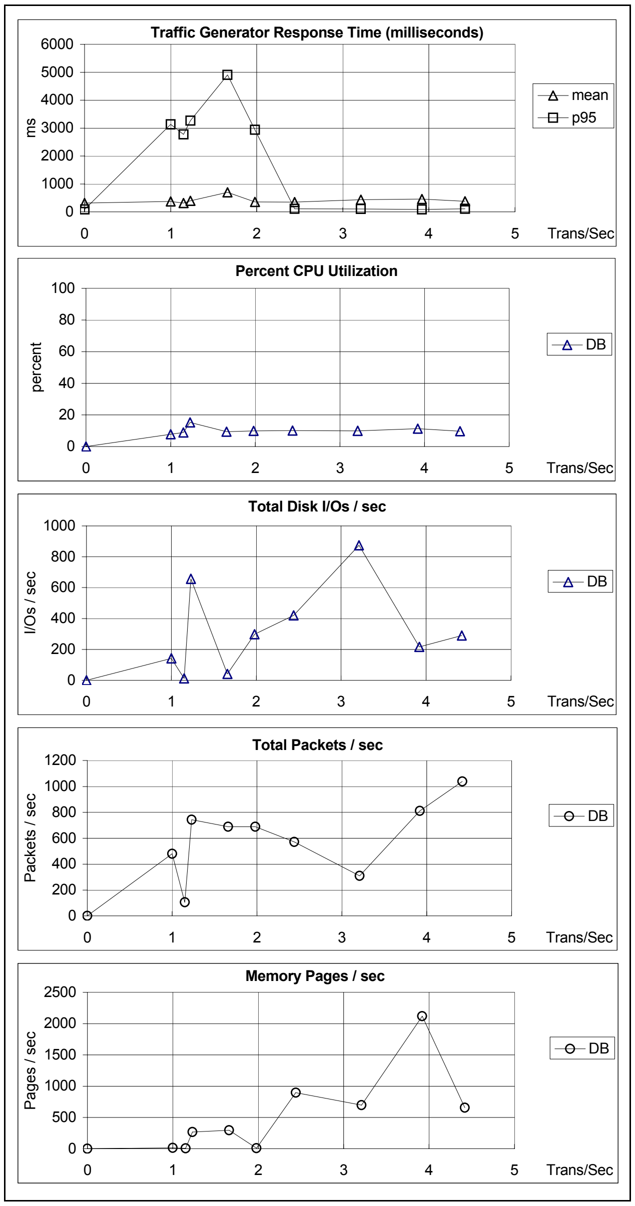
AP1: CPU, Disk I/O, Packets, Memory

run	tps	rt mean	rt p95	cpu sys	cpu tot	dsk r/s	dsk rw/s	pkt r/s	pkt rs/s	mem pg/s
1	0.00	315	86	0	0	0	0	0	0	0
2	1.00	372	3133	1	1	0	4	30	51	0
3	1.15	315	2778	1	1	0	4	21	37	0
4	1.23	398	3269	1	2	0	4	103	194	0
5	1.66	701	4908	1	2	0	5	47	80	0
6	1.98	356	2940	1	1	0	4	15	27	0
7	2.44	351	112	1	2	0	5	68	116	0
8	3.21	437	107	1	1	0	3	14	23	0
9	3.92	462	86	2	3	0	6	110	186	0
10	4.42	383	109	1	1	0	3	16	25	0

AP2: CPU, Disk I/O, Packets, Memory

run	tps	rt mean	rt p95	cpu sys	cpu tot	dsk r/s	dsk rw/s	pkt r/s	pkt rs/s	mem pg/s
1	0.00	315	86	0	0	0	0	0	0	0
2	1.00	372	3133	1	1	0	2	17	29	0
3	1.15	315	2778	1	1	0	3	51	86	0
4	1.23	398	3269	1	1	0	2	15	27	0
5	1.66	701	4908	1	1	0	2	4	7	0
6	1.98	356	2940	1	2	0	4	53	92	0
7	2.44	351	112	1	1	0	2	8	14	0
8	3.21	437	107	1	2	0	4	81	140	0
9	3.92	462	86	1	1	0	2	9	15	0
10	4.42	383	109	1	3	0	5	120	208	0

Figure 6: Database Server (DB)



DB: CPU, Disk I/O, Packets, Memory

run	tps	rt mean	rt p95	cpu sys	cpu tot	dsk r/s	dsk rw/s	pkt r/s	pkt rs/s	mem pg/s
1	0.00	315	86	0	0	0	0	0	0	0
2	1.00	372	3133	1	8	80	142	211	481	12
3	1.15	315	2778	1	9	4	13	49	106	8
4	1.23	398	3269	2	15	426	657	312	743	268
5	1.66	701	4908	2	9	34	42	273	688	296
6	1.98	356	2940	1	10	186	298	291	689	9
7	2.44	351	112	4	10	79	420	285	571	898
8	3.21	437	107	4	10	290	875	155	309	695
9	3.92	462	86	3	11	200	216	407	812	2120
10	4.42	383	109	2	10	133	290	414	1039	657

5. Lessons Learned

The limited testing time and simplified traffic mix provided many challenges to producing a high quality traffic capacity study but these circumstances presented an opportunity to learn some valuable lessons and gain important insights. The following are some of these insights.

1. This experience shows that a small initial test can isolate major system bottlenecks and help focus tuning efforts. There is a tendency to perform one large traffic capacity test of a new application just prior to release into production, but many times the most effective approach is to test incrementally and increase test complexity gradually. To that end twenty new queries and several database updates have been identified, planned, and in some cases set up for the next round of testing.
2. Limit traffic generation sources to those whose traffic mix is manageable and whose data is quantifiable. The inclusion of the “hands on” users may have added a degree of realism to this test but it introduced transactions of unknown type and volume, negatively impacted the quality of information contained in the X-Y Plots produced.
3. Compare server configuration information with system specifications before testing begins. This initial round of testing could have focused on the next bottleneck or imbalance if that step had been taken because the Database Server specification called for 16 GB of RAM but only 3.25 GB were configured.
4. Perform initial capacity tests connected directly to the server complex, e.g. Figure 1. This step allows a balanced and tuned target environment to be established before network connection complexities are introduced. During testing, the “hands on” users complained the network was slow but the traffic generator isolated the source of the long latency to the server complex because it did not have an external network connection.

6. Summary

A common server configuration that has emerged for supporting Web based applications is the Web Server, Application Server, and Database Server arrangement. This paper discusses a specific traffic capacity testing experience with this server combination for a Windows environment using a transaction oriented load generator instead of a traditional virtual user script based tool.

The paper summarizes the initial traffic capacity testing of a new application and includes the following items as part of the analysis performed and results obtained:

- A Traffic Generation Topology Diagram, Figure 1, showing the major test components and the topological relationship that exists between the traffic generator and its target resources.
- A System Configuration Table, Table 1, containing CPU, Memory, and Operating System specifics for the traffic generator as well as the target servers.
- A Traffic Mix Table, Table 2, which identifies the transactions requested along with their type, frequency, and complexity.

- A Maximum Traffic Summary Stats Table, Table 4, that lists maximum transaction rate achieved, resource consumption levels attained, and response time service levels observed.
- A Distribution of Response Times Graph, Figure 2, used to describe response time diversity.
- Traffic Generator Resource Consumption and Response Time Service Level graphs with associated tables, Figure 3. The graphs are X-Y Plots of resource consumption as a function of transaction rate and the tables contain arrival pattern statistics along with the data used to construct the graphs.
- A set of Web Server (Figure 4), Application Server (Figure 5), and Database Server (Figure 6) Resource Consumption graphs and tables.

As testing progresses and bottlenecks are removed, results reports will include:

- A CPU Utilization Process Proportion Pie Chart.
- An estimate of users supported.

This test identified a memory bottleneck in the Database Server leading to a recommendation that its memory be increased to 16 GB of RAM. Subsequent traffic capacity tests will either isolate more bottlenecks or determine that the system is balanced, tuned, and ready for production. Production resource consumption data will be correlated with the test data gathered and used for capacity projection update purposes.

7. References

[ALLE78] A.O. Allen, “Probability, Statistics, And Queueing Theory”, Academic Press, Inc., Orlando, Florida, (1978).

[BRAD04], J.F.Brady, “Traffic Generation Concepts – Random Arrivals”, www.perfdynamics.com, Classes, Supplements. (2004).

[BRAD06], J.F.Brady, “Traffic Generation and Unix/Linux System Traffic Capacity Analysis”, Journal of Computer Resource Management, CMG Journal #117:12-20, (Spring 2006).

[GUN00] N.J. Gunther, “The Practical Performance Analyst”, iUniverse Press, Lincoln, Nebraska, 2nd edition, (2000).

[JECK04] Mario Jeckle and Erik Wilde, “Identical Principles, Higher Layers: Modeling Web Services as Protocol Stack” www.idealliance.org/papers/dx_xmle04/papers/03-05-04/03-05-04.pdf , Proceedings by deepX Ltd, (2004).

[KLEI75] L. Kleinrock, “Queueing Systems Volume 1 and 2”, John Wiley & Sons, New York, N.Y., (1975).

8. Trademarks

Windows is a registered trademark of Microsoft Corporation in the U.S. and certain other countries.

Cisco is a registered trademark of Cisco Systems Corporation in the U.S. and certain other countries.

F5 is a registered trademark of F5 Networks Corporation in the U.S. and certain other countries.

Appendix A

Introduction

The purpose of this appendix is to provide a brief example of transaction based traffic generation and analysis using a target server which, unlike the servers in this paper, achieves high resource utilization levels. The example contains traffic generator output from one of the runs that produced data for both the set of X-Y Plots shown and the CPU utilization process level pie chart included.

Traffic Generator

Figure A.1 is an example run screen from the traffic generator used to produce the results contained in this paper. This particular example depicts a twelve minute run where good, bad, and late response statistics are reported every 100 seconds for both GET and POST Web queries. There are 50 GET and 50 POST queries producing requests randomly at an aggregate rate of a little over 22 transactions per second. Some late events are being recorded at this transaction rate for the three second threshold specified.

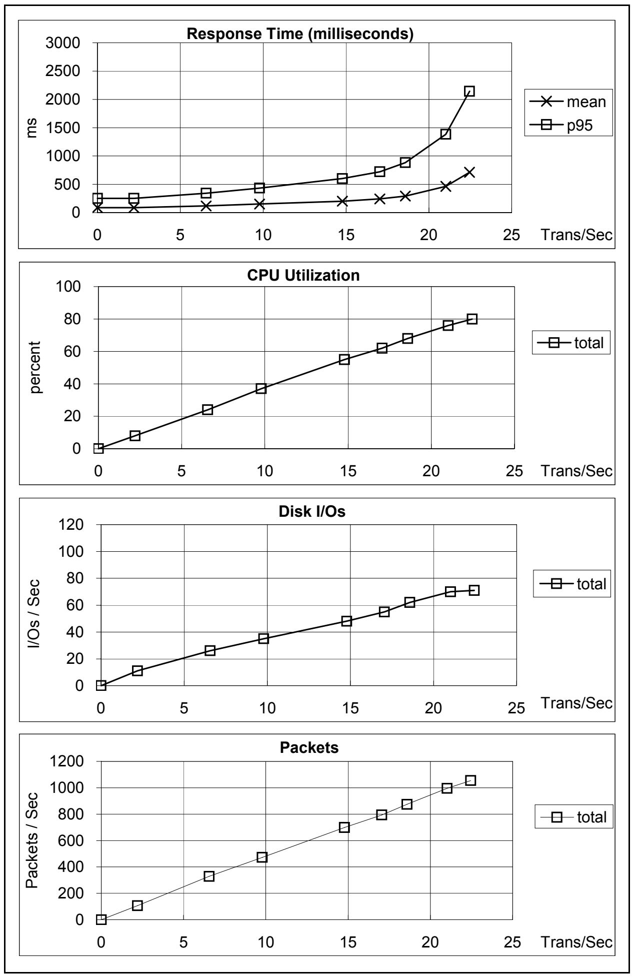
Figure A.1 Traffic Generator Run

```
Linux Sat May 6 13:53:51 2006, 50 GET and 50 POST source(s), 3000 ms late
  ./web_traffic -t -p 0085 -i 100 -s 720 3789 perf_measure_100

                               -------------GET-------------  ------------POST------------
    Time     sent    !recv    good     bad     late    good     bad     late
13:53:51     2258        0    1126       0       24    1132       0       17
13:55:31     2286        0    1147       0       15    1139       0       13
13:57:11     2204        0    1142       0       12    1062       0       17
13:58:51     2239        0    1098       0       21    1141       0        9
14:00:31     2173        0    1071       0       25    1102       0       13
14:02:11     2249        0    1128       0       23    1121       0       15
14:03:51     2334        0    1180       0       12    1154       0       14
   Total    15743        0    7892       0      132    7851       0       98

Caught a SIGALRM signal -- shutting down

Sat May 6 14:04:26 2006
```

X-Y Plots

Often the best way to show a system's capacity characteristics is to draw a picture of resource consumption levels and response time service levels as a function of incremental increases in traffic volume. The X-Y Plots and associated data table in Figure A.2 are an example of such a graphical illustration. When the transaction mix is held constant and traffic is added incrementally, as is done in Figure A.2, certain resource consumption statistics tend to increase proportional to increases in traffic volume unless there is a system imbalance or bottleneck. The basic idea is that if the workload is consistent, and doubles, the resource utilization level also doubles. The three most important resources to consider with this traffic congestion characteristic are CPUs, Disks, and Enet communications devices.

Figure A.2: Traffic Capacity – X-Y Plots



Arrival, Response Time, CPU, Disk I/O and Packets

run	tps	arr mean	arr sdev	rt mean	rt p95	cpu tot	dsk rw/s	pkt rs/s
1	0.00	0	0	88	253	0	0	0
2	2.20	453	457	88	253	8	11	107
3	6.56	151	157	116	343	24	26	328
4	9.78	101	99	150	434	37	35	473
5	14.78	67	67	203	603	55	48	698
6	17.04	58	58	242	723	62	55	794
7	18.58	53	53	291	883	68	62	875
8	21.02	47	47	464	1383	76	70	996
9	22.45	44	44	712	2143	80	71	1055

The CPU % Utilization, Disk I/O rate, and Enet Packet rate graphs in Figure A.2 exemplify throughput proportionality since consumption levels for all of them are a linear function of traffic rate during the eight traffic runs performed. It appears from this graph that the system is approaching a CPU limitation at around 80% Utilization because both the mean and p95 response times are increasing sharply. Disk and Enet are unlikely to be causing this service level degradation since their traffic rates are both below normal saturation levels and their X-Y Plots are linear to the end.

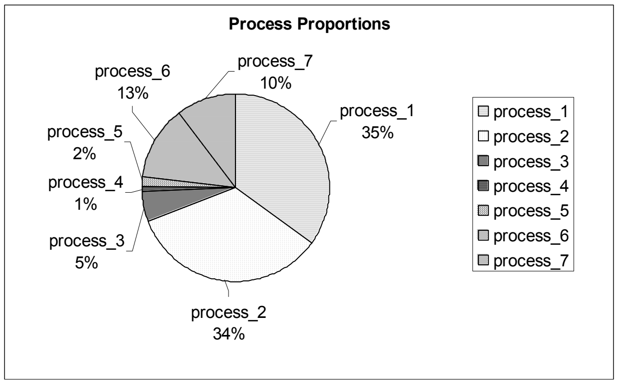
The traffic generator output in Figure A.1 represents the last observation in the Figure A.2 set of X-Y Plots when CPU Utilization is 80% and the p95 response time is 2.143 seconds.

There are some system resources, like memory, where throughput proportioning does not work. This is because memory allocation isn't based on transaction rate but on number of running processes and associated active threads. As shown in Figure 6, traffic volume impacts paging activity, but not in a proportional way.

Process Proportions

The Figure A.3 pie chart and associated table expand upon the Figure A.2 CPU Utilization Graph by indicating which processes use CPU time and to what degree. This graph is created by proportioning the CPU time recorded by the Windows PerfMon templates during the last four traffic runs. It is particularly helpful to software developers for focusing their tuning efforts on processes which consume the most CPU time. There is little payoff improving the performance of a process which uses 1% of the CPU by 50% but such a performance improvement on a process that uses 35% is significant.

Figure A.3: Process Proportions - Pie Chart



	Processor Seconds In 720 Second Run					
Name	**Run 6**	**Run 7**	**Run 8**	**Run 9**	**Total**	**Percent**
process_1	**156**	**172**	**192**	**202**	**722**	**35%**
process_2	**152**	**167**	**186**	**196**	**700**	**34%**
process_3	**23**	**26**	**29**	**30**	**108**	**5%**
process_4	**5**	**5**	**6**	**6**	**23**	**1%**
process_5	**7**	**7**	**8**	**9**	**31**	**2%**
process_6	**58**	**63**	**71**	**75**	**266**	**13%**
process_7	**45**	**50**	**56**	**58**	**209**	**10%**
total	**446**	**490**	**547**	**576**	**2059**	**100%**